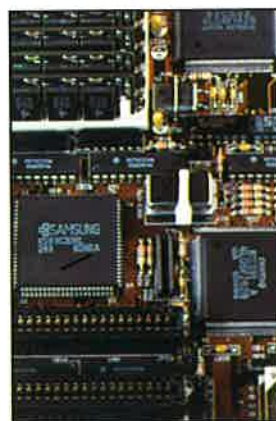
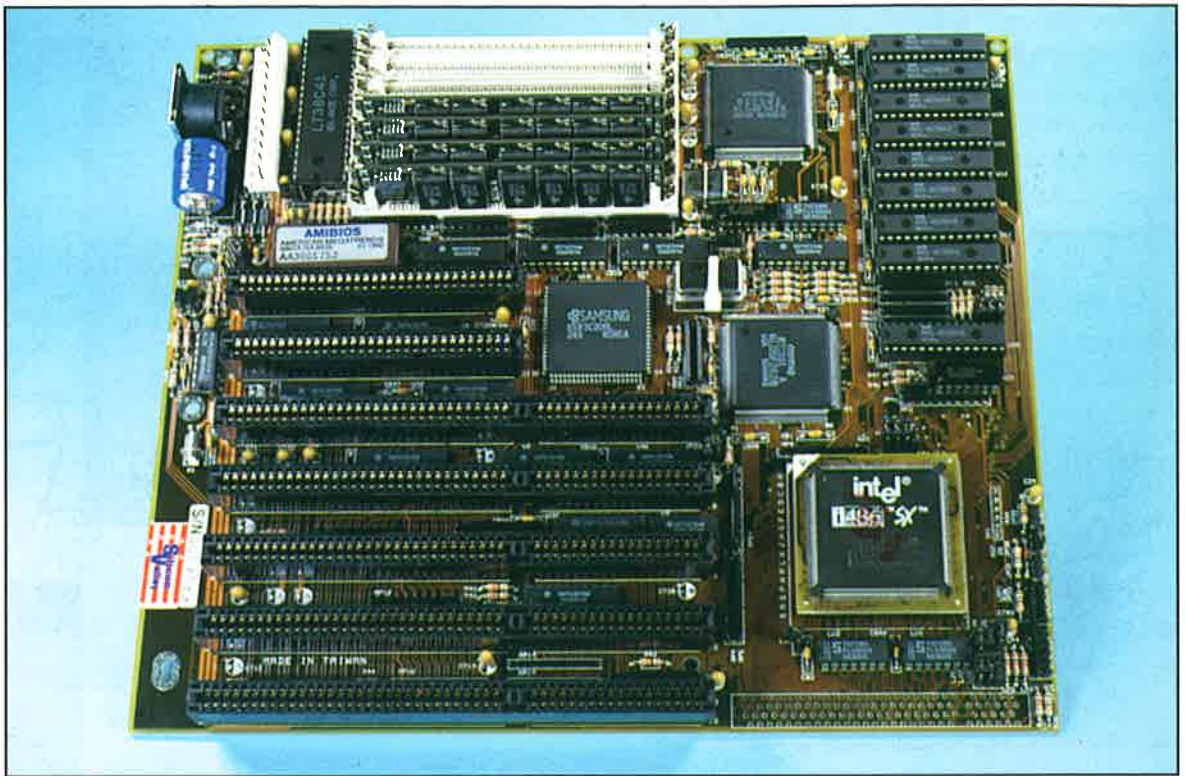
# IL 486

**Quando Intel ha pianificato lo sviluppo del 486, ha proseguito sulla strada che aveva portato al successo gli altri microprocessori della famiglia. La filosofia seguita dai suoi progettisti è stata quella dell'evoluzione derivata dall'avere mantenuto tutto ciò che in precedenza era già stato creato. Questa filosofia si può riassumere in quattro parole: la compatibilità innanzi tutto.**

L'investimento fatto da Intel per lo sviluppo del microprocessore 80486 è stato di 250 milioni di dollari. Questa enorme cifra è stata sicuramente ben spesa, soprattutto considerando il crescente mercato dei microprocessori, che nel 1991 ha raggiunto un fatturato di circa 4.000 milioni di dollari. Un'altra ragione per cui Intel ha proseguito nei suoi investimenti e nei suoi progetti a tappe forzate è stata quella di voler dettare gli standard di mercato. Infatti, la società che per prima riesce a lanciare un prodotto sul mercato è in grado di dettare anche le regole, imponendo la propria tecnologia che i

*L'investimento economico fatto da Intel per sviluppare il microprocessore 486 è stato di 250 milioni di dollari*

*Il microprocessore 486 contiene sul suo wafer di silicio 1,2 milioni di transistor in tecnologia CMOS*

*Tipica scheda madre 486*

concorrenti, per stare al passo, sono costretti a seguire. Fino ad oggi Intel è stata in grado di poter lavorare con successo ed in modo autonomo, ma il crescente numero di competitori e l'impegno economico e umano richiesto per la ricerca e lo sviluppo di microprocessori ancora più potenti sono ormai troppo gravosi per un solo costruttore; e quindi è ormai prossimo il momento in cui le diverse società produttrici saranno costrette a collaborare tra di loro e creare alleanze, come è già successo in molti altri campi.



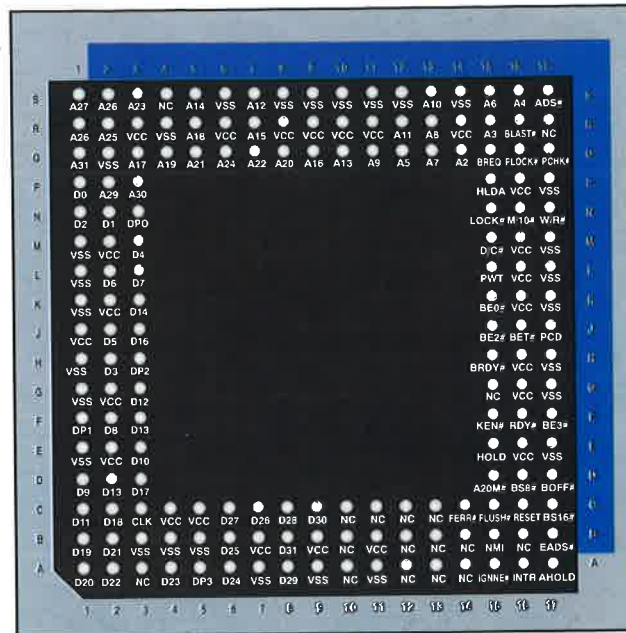
*Struttura tipica di un microprocessore 80486*

## CARATTERISTICHE PRINCIPALI

Il microprocessore 486 presenta le caratteristiche migliori con applicazioni DOS, OS/2, WINDOWS e UNIX System V/386, ed è completamente compatibile a livello binario con i microprocessori 386DX e 386SX. Il suo chip di silicio contiene 1,2 milioni di transistor in tecnologia *CMOS* (*Complementary Metal-Oxide-Semiconductor*) con dimensioni di 1 micron (per i modelli a 25/33 MHz) e da 0,8 micron per il modello a 50 MHz (un micron equivale a 0,000001 metri). L'insieme di questi transistor viene utilizzato per la memoria cache, per il coprocessore matematico, per il gestore della memoria e per altri sottosistemi. Una elevata percentuale delle sue istruzioni sono di tipo *RISC* (*Reduced Instructions Set Computing*, o gruppo di istruzioni ridotte). Si caratterizza per una memoria cache integrata da 8 Kbyte, unificata per codici e dati, con un bus che può lavorare a 25/33 MHz trasferendo dati ad una velocità di 80/106 Mbyte/s; ciò garantisce un alto rendimento al sistema anche se è dotato di memorie a basso costo come le *DRAM* (*Dynamic RAM*).

Inoltre, presenta caratteristiche nuove per operazioni in multiutenza, una gestione della memoria basata su di un sistema semaforico, una memoria cache multilivello, ecc. Per quanto concerne la velocità, a 25 MHz esegue 20 ***MIPS*** (*milioni di istruzioni al secondo*), a 33 MHz questo valore sale a 27 MIPS, mentre a 50 MHz raggiunge la considerevole quantità di 41 MIPS.
Alcune delle principali caratteristiche del 486 sono:

*- compatibilità binaria completa con le CPU 386DX, 386SX, coprocessore 387, 80286, 8086 e 8088,*
*- unità di esecuzione progettata per eseguire con un solo ciclo di clock le istruzioni più comunemente utilizzate (tecnica RISC),*
*- processore per lo sviluppo di operazioni aritmetico-logiche da 32 bit,*
*- unità aritmetica a virgola mobile (coprocessore matematico) interna, che supporta formati da 80, 64 e 32 bit specifici per lo standard 754 dell'IEEE (essendo il codice oggetto compatibile con il 387DX e SX),*
*- memoria cache interna da 8 Kbyte che permette un accesso veloce alle istruzioni e ai dati usati più di recente,*
*- segnali di controllo del bus per mantenere immutata la cache in sistemi a multiprocessore (diversi processori che operano in parallelo),*
*- possibilità di segmentazione, un sistema di gestione della memoria per creare spazi di indirizzamento protetti e indipendenti,*
*- possibilità di paginazione, un sistema di gestione della memoria che consente di accedere a strutture di dati di dimensioni maggiori dello spazio di*

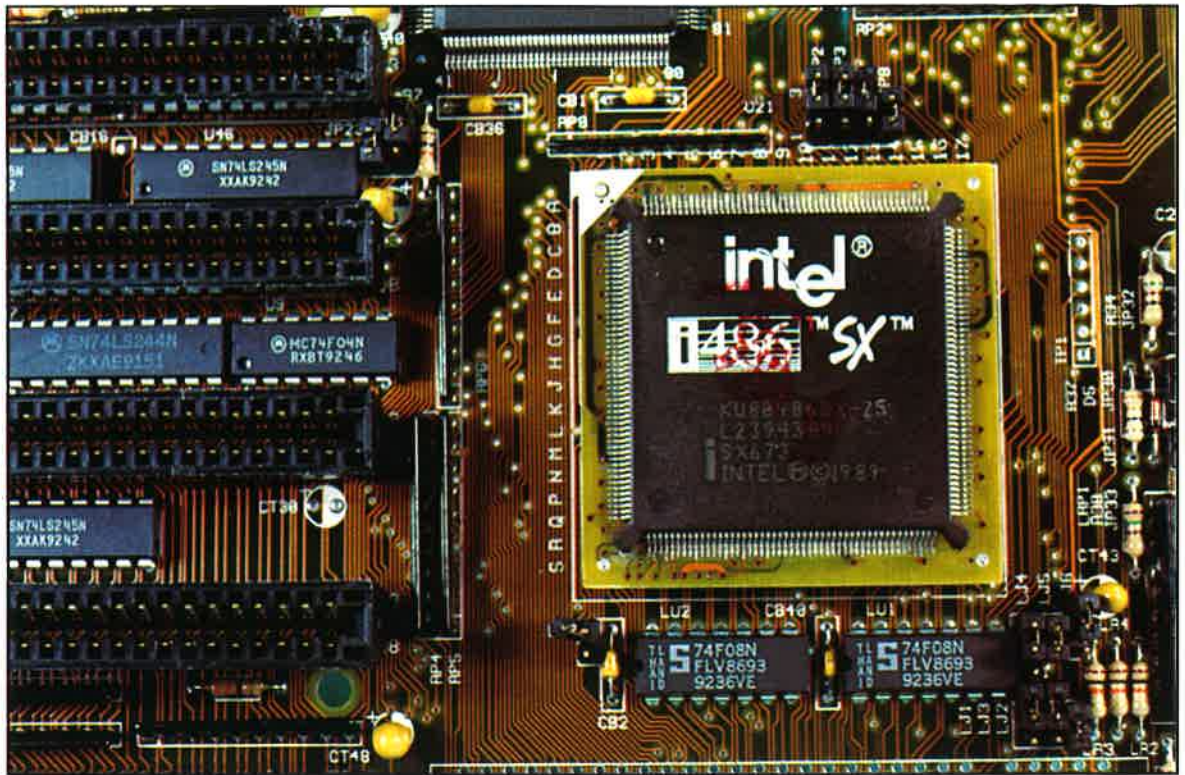
*Registri a 32 bit di un 486: di uso generale, di segmento, di controllo e di stato*

*Disposizione dei terminali di un microprocessore 486 in configurazione Pin-Grid-Array*

*memoria disponibile, mantenendo parte di essi in memoria e parte sul disco,*
*- istruzioni di avvio che, tramite una eccezione (necessaria per supportare la memoria virtuale paginata), consentono di lanciare direttamente una applicazione*
*- possibilità di eseguire istruzioni in cascata (Pipeline), in modo che vi sia una differenza tra le diverse istruzioni,*
*- presenza di registri per la depurazione del codice e dei dati supportati dall'hardware.*

Inoltre il microprocessore 486 è compatibile a livello di codice oggetto con i tre microprocessori della famiglia 386:
- con il microprocessore 386DX (bus dei dati a 32 bit), utilizzato in personal computer di fascia alta e in stazioni di lavoro di fascia media,
- con il microprocessore 386SX (bus dei dati a 16 bit), utilizzato in personal computer di fascia media di minor costo rispetto a quelli con bus a 32 bit,
- con il microprocessore industriale 376 (bus dei dati a 16 bit), che costituisce una forma ridotta ottimizzata del 386 per applicazioni industriali come controller di processo.

La modalità di funzionamento del microprocessore 486 determina le caratteristiche dell'architettura e le istruzioni disponibili per questa modalità. Il microprocessore è in grado di operare in tre modalità di funzionamento:
- *modalità protetta*, nella quale riconosce il set

*Il microprocessore 486 è dotato di una memoria cache interna di 8 Kbyte*

*Il reset iniziale imposta il microprocessore nella modalità di indirizzamento reale*

*I modelli SX della famiglia 486 hanno la particolarità di non avere il coprocessore matematico integrato*

originale delle istruzioni a 32 bit. In questa modalità sono disponibili tutte le istruzioni e possono essere sfruttate completamente le caratteristiche del 486,

- *modalità di indirizzamento reale* (anche chiamata modalità reale), nella quale viene emulato l'ambiente di lavoro del microprocessore 8086. Il "reset" di avvio imposta il microprocessore in questa modalità operativa,
- *modalità virtuale 8086* (anche chiamata modalità V86), che costituisce un altro sistema di emulazione dell'8086. A differenza della modalità reale, quest'ultima è compatibile con la protezione e la gestione della memoria. Dalla modalità protetta è possibile passare alla modalità V86, eseguire un programma scritto per l'8086, e tornare alla modalità protetta sfruttando il set di istruzioni a 32 bit.

*Registri del sottosistema a virgola mobile*

## Sottosistemi del 486

All'interno di un microprocessore 486 si possono distinguere 9 unità funzionali diverse, o sottosistemi. Ciascuno di questi può funzionare in parallelo con gli altri sulla stessa istruzione o su istruzioni diverse:

**1.** *unità di interconnessione con il bus*; è l'unico sottosistema che comunica con il mondo esterno al microprocessore. Gli altri sottosistemi lo utilizzano per accedere alla memoria principale, ai dischi, e alle porte di I/O (Ingresso/Uscita). Internamente comunica con l'unità di prericerca e con la cache, alla quale accede con un bus dati e un bus indirizzi entrambi a 32 bit.
Una delle funzioni svolte da questo sottosistema è quella di inviare il codice e i dati della memoria principale, quando nella cache questi non sono presenti, al sottosistema di prericerca e alla cache. Il bus indirizzi a 32 bit che collega questa unità alla cache è bidirezionale, e serve per indirizzare

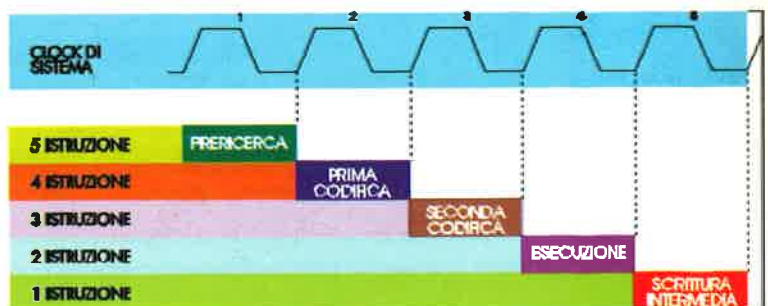
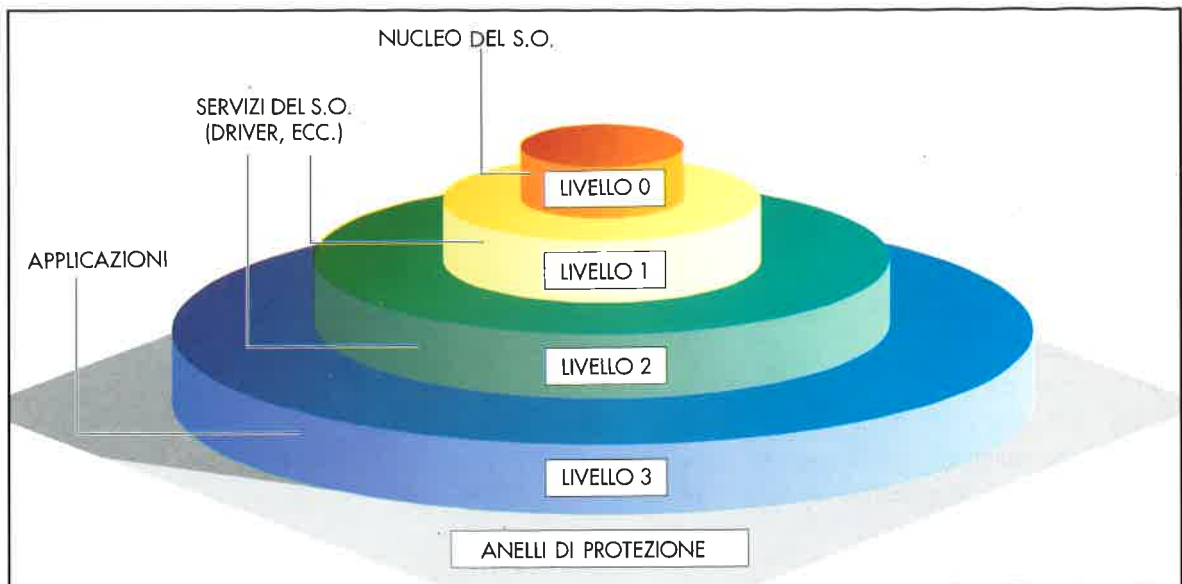
*Nello schema sono evidenziati i nove sottosistemi integrati in un microprocessore 486*

La decodifica viene realizzata in due stadi che richiedono due cicli di clock

la memoria principale e le locazioni della cache che devono essere cancellate in seguito a una riscrittura della memoria principale da parte di qualche dispositivo esterno.
Questo primo sottosistema è composto da altre otto sottounità:
- *transcettori di indirizzi*. Le linee A2..A31 servono il bus indirizzi esterno, e i bit degli indirizzi A4...A31 sono bidirezionali per convalidare il contenuto della cache,
- *registri intermedi di scrittura*, costituiti da quattro buffer da 80 bit che servono per trattenere momentaneamente i dati e gli indirizzi che devono essere scritti sul bus esterno, in modo da non ritardare il lavoro del processore ma impegnarlo per questa operazione solo quando è libero,
- *transcettori dei dati*, costituito dal bus a 32 bit (D0..D31) sul quale avviene l'invio e la ricezione dei dati,
- *controller del bus*, che attiva i segnali di controllo che servono per amministrare i diversi cicli del bus,
- *controller di compattazione*, che permette il trasferimento fino a 16 ottetti per ogni ciclo di clock,
- *controller della dimensione del bus*, che modifica la dimensione dello stesso a 32, 16 o a 8 bit in funzione del tipo di periferica alla quale i dati vengono inviati,
- *controller della cache*, che verifica se un dato è memorizzato nella cache oppure no tramite un segnale esterno di controllo,
- *controllo e generazione della parità*. Il microprocessore genera un bit di parità per ogni ottetto scritto, che viene verificato nel momento in cui l'ottetto stesso viene letto.

**2.** *unità di prericerca*; fornisce le istruzioni che devono essere eseguite dall'unità di decodifica. Comunica con la cache tramite un bus a 128 bit e con l'unità di segmentazione tramite un bus a 32 bit. Questa unità genera cicli di bus per il caricamento delle istruzioni nei suoi registri.

**3.** *sottosistema di decodifica*; interpreta le istruzioni generando i segnali di controllo necessari e i blocchi delle microistruzioni. Opera alla frequenza di una istruzione decodificata per ogni ciclo di clock. La decodifica avviene in due fasi:
- prima fase: inizializzazione del ciclo del bus per l'esecuzione dell'istruzione di accesso,
- seconda fase: elaborazione dei dati che hanno

*Schema delle fasi di una elaborazione in cascata (Pipeline)*

*Le operazioni semplici vengono eseguite generalmente in un solo ciclo di clock*

*Osservando gli anelli relativi ai diversi livelli di protezione, si può notare che il grado più alto corrisponde al nucleo del sistema operativo*

avuto accesso; queste due fasi utilizzano due cicli di clock.

**4.** *unità di controllo*; esegue le microistruzioni ricevute dall'unità di decodifica e definisce le operazioni che devono essere effettuate sui numeri interi, a virgola mobile o sul sistema di segmentazione. Questa unità è dotata di una ROM contenente un microprogramma che verifica gli attributi di protezione, gestisce le eccezioni generate dall'unità a virgola mobile, ed esegue la rilevazione degli interrupt.

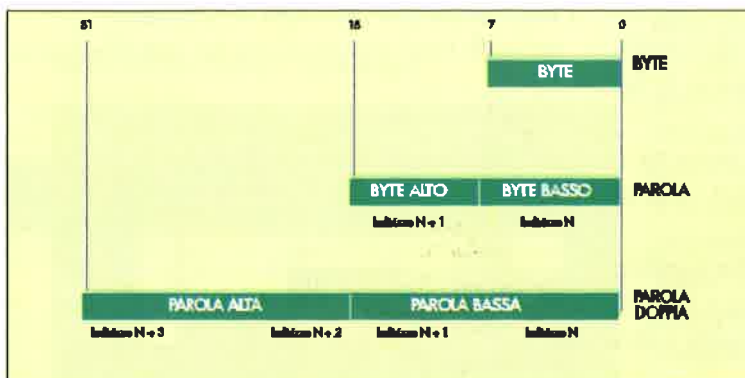
**5.** *unità degli interi*; ha il compito di eseguire i calcoli aritmetico-logici nell'intervallo numerico permesso dal microprocessore. Questa unità è composta dai seguenti elementi:

- unità aritmetico-logica(ALU),
- unità di scorrimento a revolver,
- banco di 8 registri per uso generale,
- indicatori di stato e condizioni,
- ROM contenente le costanti più utilizzate.

Le operazioni semplici, quali somme, sottrazioni o caricamento dei registri, vengono normalmente eseguite in un solo ciclo di clock.

Internamente questo sottosistema è dotato di cinque bus a 32 bit per il collegamento dei registri.

**6.** *unità a virgola mobile*; le operazioni a virgola mobile vengono eseguite da questo sottosistema, che internamente è compatibile con il 387 anche se risulta da 3 a 5 volte più veloce. Il calcolo dei numeri reali può essere realizzato con formati da 32, 64 e 80 bit. Quando questa unità deve eseguire calcoli molto lunghi e complessi, il suo sequenziatore indipendente gli consente di funzionare mentre l'unità degli interi continua ad elaborare le istruzioni.

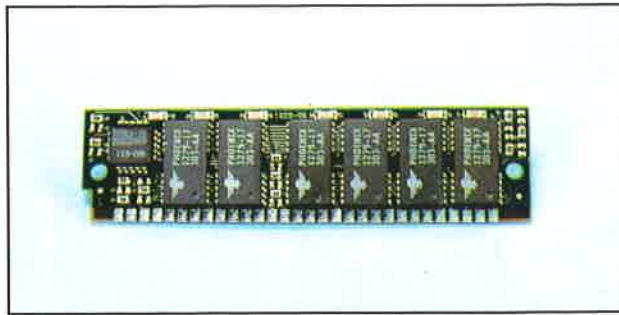
**7.** *unità di segmentazione*; ogni operazione può avere sino a 16.383 segmenti. Nei descrittori si trova l'indirizzo di partenza del segmento, i suoi limiti, i suoi attributi e la sua priorità. L'indirizzo di partenza può essere uno qualsiasi compreso nei 4 Gbyte consentiti dalla memoria fisica. L'indirizzo

*Tipi di dati fondamentali e loro indirizzamento generico in un processore a 32 bit*

virtuale, che è l'indirizzo del programma basato sui segmenti, è costituito da 46 bit che permettono un indirizzamento di 64 Tbyte (terabyte).

**8.** *unità di paginazione*; se si attiva questa unità lo spazio di indirizzamento lineare viene suddiviso in pagine da 4 Kbyte. In questo modo possono essere eseguiti programmi molto più ampi della memoria disponibile, poiché una parte degli stessi viene caricata in memoria mentre la parte restante rimane sul disco.

**9.** *unità cache*; è condivisa sia dai dati che dagli indirizzi. La sua struttura è di tipo associativo a quattro vie, con linee da 16 bit ciascuna. La cache memorizza i dati associandoli ai loro indirizzi fisici e non agli indirizzi virtuali.

## Categorie di dati

I dati gestiti dal 486 possono essere suddivisi in tre grandi categorie che hanno le seguenti forme e lunghezze: il byte (8 bit), la parola (16 bit) e la parola doppia (32 bit). A loro volta queste categorie comprendono i seguenti tipi di dati:
*intero:* è un numero binario, dotato di segno che viene sempre indicato dal bit più significativo (bit 7 in un byte, bit 15 in una parola, e bit 31 in una parola doppia),
*ordinale:* numero binario senza segnò, che può essere come il precedente da 8, 16 o 32 bit,
*puntatore prossimo:* è un registro logico da 32 bit utilizzato normalmente come puntatore per indirizzi all'interno di un segmento,
*puntatore remoto:* registro logico da 48 bit costituito da un selettore di segmento a 16 bit e da un puntatore a 32 bit,
*stringa:* sequenza continua di byte, parole o doppie parole fino ad un massimo di 4 Gbyte,
*campo di bit:* sequenza continua di bit che può iniziare in una qualunque posizione di qualsiasi byte ed avere una lunghezza fino a 32 bit,
*stringa di bit:* sequenza continua di bit che può iniziare in una qualsiasi posizione di qualunque byte ed avere una lunghezza fino a 4 Gbyte,
*BCD non compattato:* ogni byte contiene una cifra decimale da 0 a 9, con il semibyte più alto generalmente impostato a 0,

*Le memorie utilizzate con il 486 sono economiche e non eccessivamente veloci, ma la loro lentezza è compensata dai sistemi cache*

*BCD compattato:* ogni byte contiene due cifre decimali da 0 a 9, una per ogni semibyte
*virgola mobile:* comprende l'insieme di tutti i numeri reali.

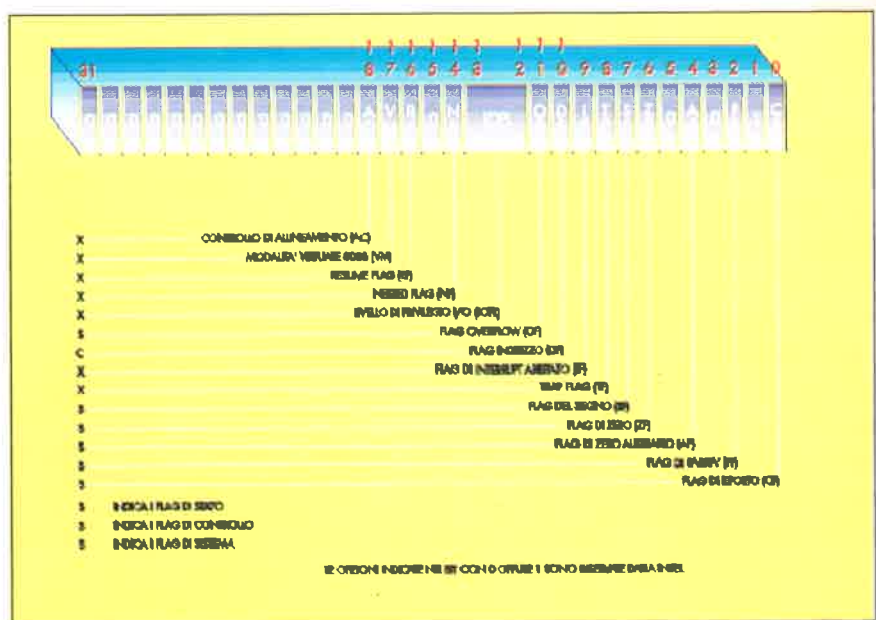
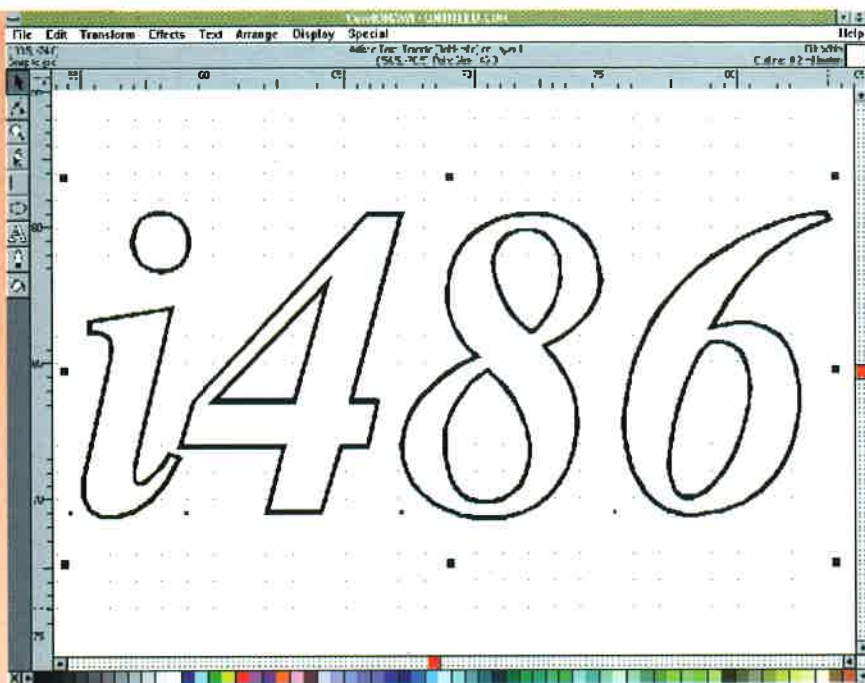
## Registri

Il 486 è dotato di un numero di registri pari al 386 e di uguale lunghezza; otto registri sono uso generale mentre sei sono i registri di segmento. La stessa cosa avviene per il coprocessore integrato. Nella cache interna del 486 sono presenti tre registri che servono per una sua verifica da parte dell'utente.

*Il 486 è dotato di 8 registri per uso generale e 6 registri di segmento*

## Tipi di istruzioni

Lo sviluppo dei processori *RISC* (*Reduced Instructions Set Computing*) ha permesso di velocizzare l'esecuzione delle istruzioni grazie al

*Registro di controllo e di stato EFLAGS con tutte le sue indicazioni*

*Nelle applicazioni grafiche è possibile notare un notevole incremento della velocità rispetto al 386*

set di istruzioni ridotto di cui fanno uso che consente il parallelismo dell'elaborazione all'interno della CPU. Anche il 486 è stato sviluppato secondo questa filosofia; infatti, è dotato di cinque stadi (pipeline a 5 stadi) che permette l'elaborazione di più istruzioni in un solo ciclo macchina. Alcune istruzioni richiedono però due cicli e altre, come i salti condizionati, ne richiedono tre. Ciò porta ad una velocità di elaborazione media di una istruzione ogni 1,8 cicli.
Grazie poi al set di istruzioni esteso, si può considerare il 486 come un microprocessore costruito con architettura RISC avvolto in una shell *CISC (Complex Instructions Set Computing)*
Di seguito viene riportata una tabella comparativa tra due modelli di processori RISC (SPARC e 88000), il 386 e il 486, nella quale vengono evidenziati i cicli di clock impiegati dai diversi dispositivi per eseguire la stessa istruzione:

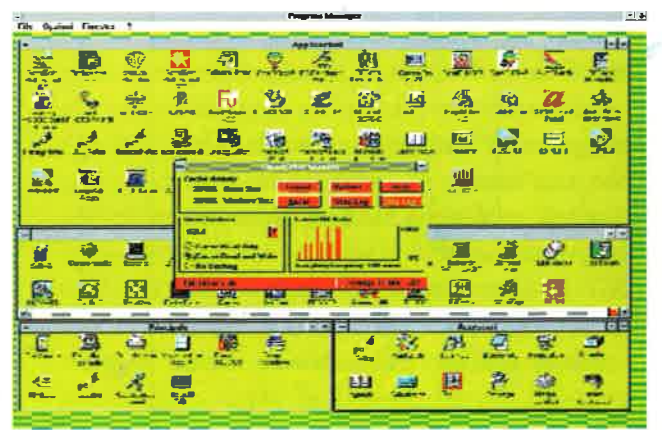
| TIPO DI ISTRUZIONE | 386 | 486 | SPARC | 88000 |
|---|---|---|---|---|
| LOAD | 4 | 1 | 2 | 1 |
| STORE | 2 | 1 | 3 | 1 |
| REG./REG. | 2 | 1 | 1 | 1 |
| JUMP(SI/NO) | 9/3 | 3/1 | 1/2 | 1 |
| CALL | 9 | 3 | 3 | 1 |

*In pratica, un programma realizzato con l'assemblatore del 486 risulta essere tra il 20 e il 40% più compatto di un programma scritto per un processore RISC*

Come si può facilmente notare, il processore che richiede più cicli di clock per eseguire una stessa istruzione è il 386.
Inoltre, è possibile affermare che il numero totale di cicli di clock richiesti dal 486 è quasi uguale al numero totale di cicli richiesti dagli altri due processori costruiti in tecnologia RISC.
In pratica però, un programma realizzato con l'assembler del 486 risulta essere del 20-40% più compatto rispetto a quello di un processore RISC

## PROTEZIONE

La protezione può essere applicata sia ai segmenti che alle pagine; l'impostazione di due bit di un particolare registro definisce il livello di privilegio del programma in corso di esecuzione.
La protezione consente di limitare il numero di interferenze che un programma difettoso può provocare, e risulta molto utile nel momento in cui è necessario "depurare" il software sviluppato.
Prima di ogni ciclo di accesso alla memoria vengono eseguiti dei controlli per verificare che i sistemi di protezione siano soddisfatti. Esistono 5 livelli di protezione:
- *controllo tipo,*
- *controllo limite,*
- *restrizione del campo indirizzabile,*
- *restrizione dei punti di ingresso ad un procedimento,*
- *restrizione di un insieme di istruzioni.*

*Il rendimento della cache non dipende dalla sua dimensione ma dagli algoritmi di gestione*

# I CODIFICATORI

**Quando si deve progettare un sistema digitale è necessario rappresentare o codificare in forma binaria l'informazione numerica e alfanumerica che deve essere gestita dal sistema stesso. Per effettuare questa trasformazione si possono utilizzare i circuiti combinatori denominati codificatori.**

Un *codificatore* è un circuito combinatorio dotato di un numero determinato di ingressi; di questi solo uno assume lo stato logico 1 in un dato istante, e genera in uscita un codice composto da diversi bit in funzione dell'ingresso eccitato. Per esprimere meglio questo concetto viene proposto un esempio. Si supponga di dover trasformare in codice binario il segnale generato dai tasti di una tastiera numerica, come può essere quella di una calcolatrice. Quest'ultima è generalmente dotata di almeno 10 tasti che rappresentano i numeri e di 6 tasti per i caratteri speciali; per semplificare l'esposizione però, si considerano solo le dieci cifre, che possono essere rappresentate in forma binaria con soli 4 bit. Inoltre, si suppone che la tastiera sia collegata in modo che la pressione di un tasto provochi la chiusura di una sola linea di ingresso; questa situazione circuitale si può osservare nella figura corrispondente.
All'interno del blocco indicato come *codificatore* si possono immaginare alcuni conduttori incrociati che uniscono ingressi e uscite tra di loro. Di seguito viene analizzato il modo con il quale devono essere internamente collegati per poter fornire in uscita i codici desiderati. Per rappresentare questi ultimi si è utilizzato un formato binario definito *codice BCD* (nel capitolo successivo verranno esaminati dettagliatamente i codici di numerazione più utilizzati nel progetto dei circuiti digitali). La

*Schema a blocchi di un codificatore a 10 ingressi e 4 uscite*

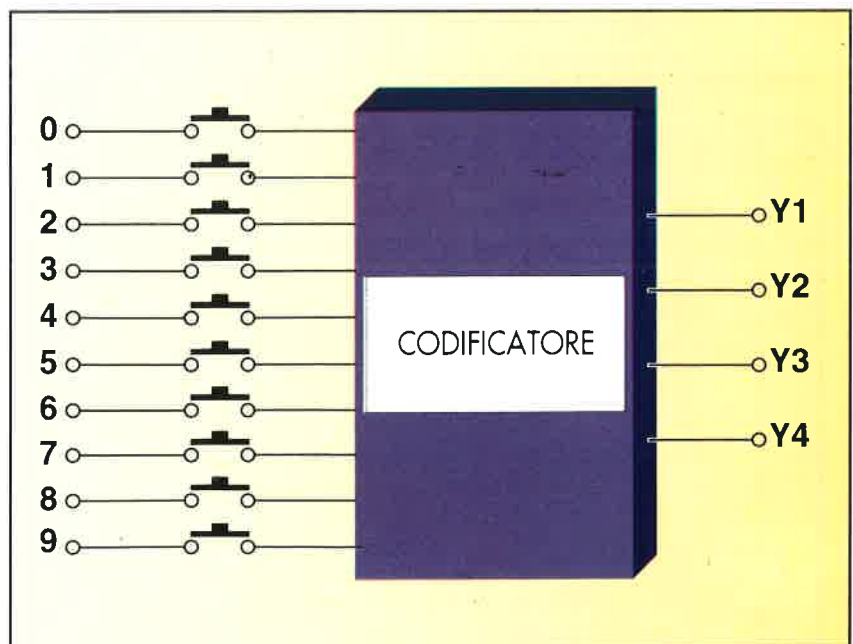
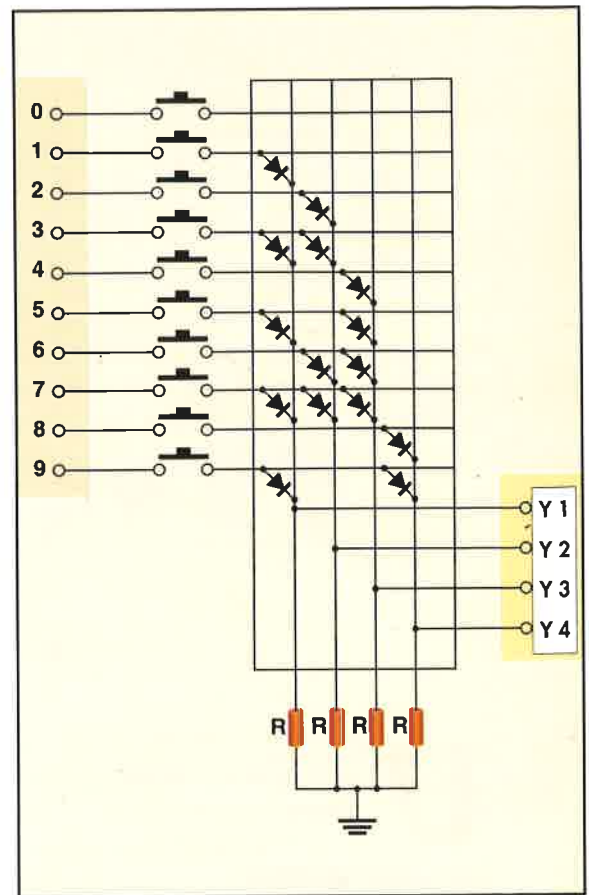
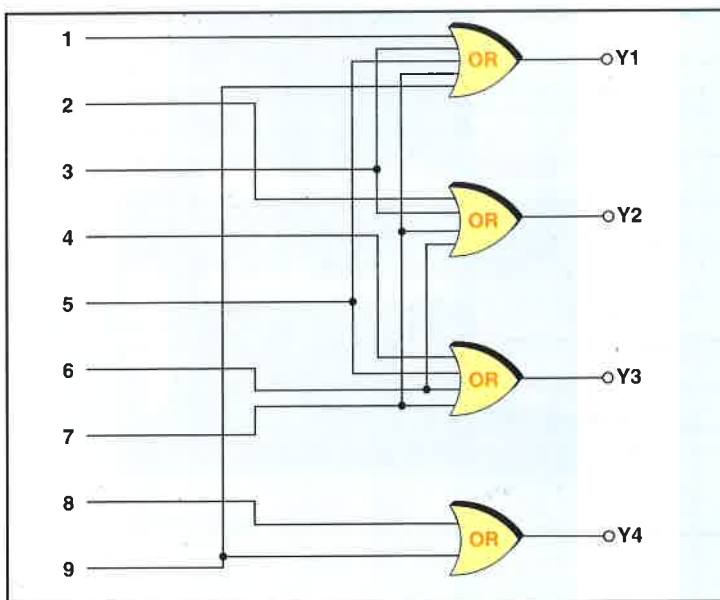
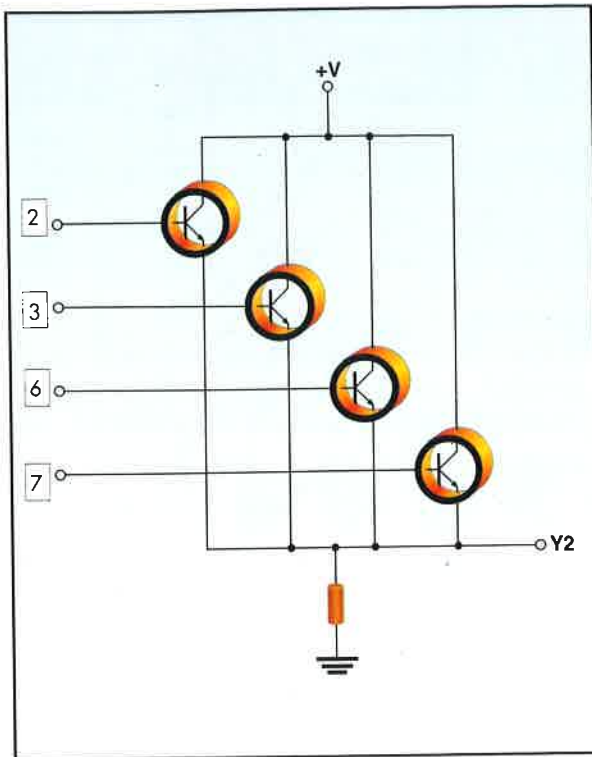
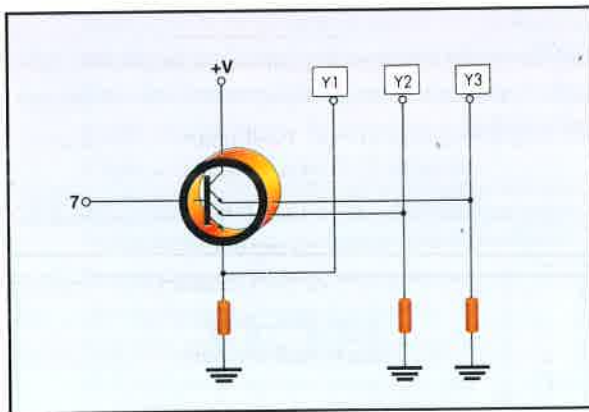



tabella della verità di questo codificatore è la seguente:

| INGRESSI | | | | | | | | | | USCITE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Y4 | Y3 | Y2 | Y1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

Premendo uno qualsiasi dei pulsanti si commuta la corrispondente linea di ingresso al livello logico 1. Se per ipotesi non è possibile premere più di un pulsante alla volta, allora tutti gli ingressi saranno a livello 0 tranne quello corrispondente al pulsante premuto in quell'istante.
Osservando la tabella della verità di questo circuito, si può facilmente vedere che l'uscita Y1 assume il valore 1 quando vengono premuti i pulsanti 9, oppure 7, oppure 5, oppure 3, oppure 1, per cui la sua equazione booleana corrispondente risulta dalla somma degli ingressi 1, 3, 5, 7 e 9. Se si prosegue l'analisi della tabella si possono ottenere con lo stesso procedimento le equazioni che devono soddisfare le uscite Y2, Y3, e Y4.
Ogni equazione ricavata dalla tabella della verità può essere rappresentata con porte logiche costruite con dei diodi. Nell'esempio proposto, il blocco codificatore che in precedenza era stato immaginato come un insieme di conduttori intrecciati tra di loro, viene di seguito rappresentato con un gruppo di diodi opportunamente collegati tra di loro; questo nuovo blocco viene definito *matrice di diodi* ed è rappresentato nella corrispondente figura.
Questi diodi possono essere sostituiti anche con dei transistor, nei quali viene sfruttata la giunzione base-emettitore; se al collettore di ogni transistor viene collegata la tensione di alimentazione si ottengono delle configurazioni ad inseguitore di emettitore che formano delle porte OR, come si può osservare nella figura corrispondente.
In pratica però, per ogni ingresso del codificatore viene utilizzato un solo transistor a emettitore multiplo. La base viene collegata ad una linea di ingresso e ogni emettitore viene collegato ad una uscita diversa, in accordo con la logica del decodificatore.
La linea di ingresso 7 ad esempio, è collegata a tre diodi i cui catodi sono collegati alle uscite Y1,

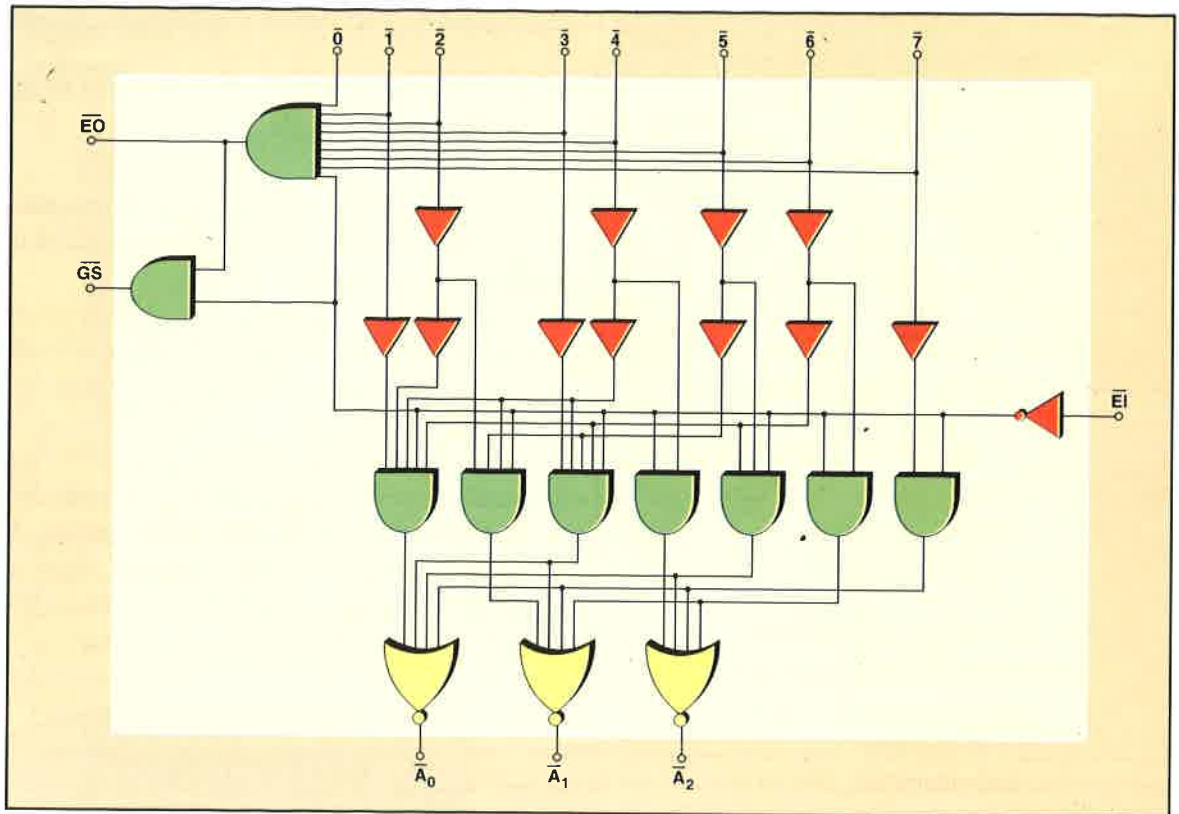
*Realizzazione di un codificatore tramite porte OR*

*I tipi di codificatori più comuni sono quelli a matrice di diodi*

*I diodi possono essere sostituiti dalla giunzione base-emettitore di un transistor*

*La struttura più classica dei codificatori a matrice di diodi prevede l'utilizzo di un transistor multiemettitore per ciascuno degli ingressi*

Y2 e Y3; questa combinazione potrebbe essere sostituita da un transistor con tre emettitori, collegato come nello schema riportato nella figura corrispondente.
Il numero massimo di emettitori richiesti è pari al numero dei bit che formano il codice di uscita. Normalmente un codificatore utilizza dei livelli TTL per cui le sue uscite possono essere di tipo *totem-pole* oppure a *collettore aperto*.
Di codificatori ne esistono di diversi tipi, come ad esempio quello descritto in precedenza a celle di diodi oppure quelli definiti *codificatori a priorità*.

## CODIFICATORI A PRIORITÀ E SEGNALE DI ABILITAZIONE

Di seguito viene abbandonata l'ipotesi precedente, per cui non era ammissibile premere più di un tasto alla volta.
Se casualmente vengono premuti contemporaneamente più tasti, è necessario assegnare un livello di priorità alla linea di ingresso dei dati di ordine superiore.
Se ad esempio vengono premuti i tasti relativi agli ingressi 5 e 6, l'uscita deve corrispondere alla combinazione definita dall'ingresso 6. Per meglio comprendere questo tipo di decodificatori, viene di seguito riportata la loro tabella della verità:

| INGRESSI | | | | | | | | | | USCITE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Y4 | Y3 | Y2 | Y1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | X | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | X | X | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | X | X | X | X | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | X | X | X | X | X | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | X | X | X | X | X | X | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | X | X | X | X | X | X | X | 0 | 1 | 1 | 1 |
| 0 | 1 | X | X | X | X | X | X | X | X | 1 | 0 | 0 | 0 |
| 1 | X | X | X | X | X | X | X | X | X | 1 | 0 | 0 | 1 |

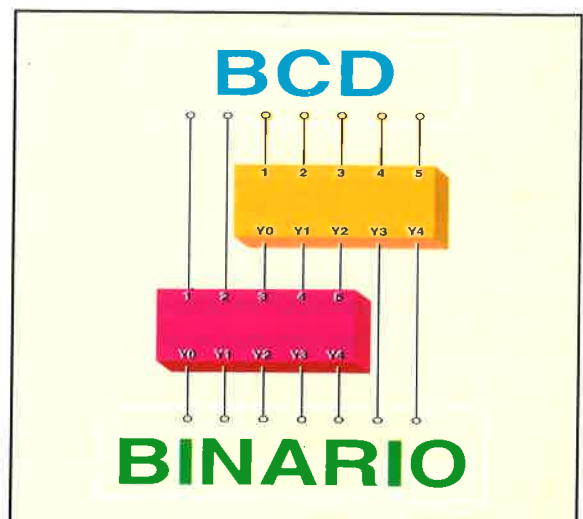
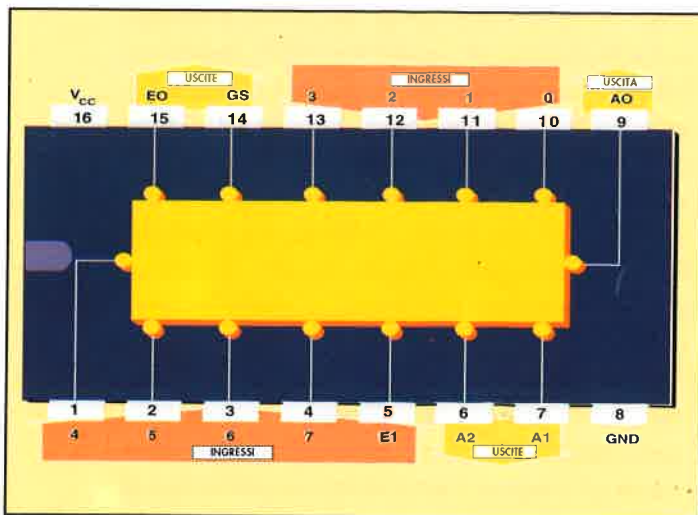
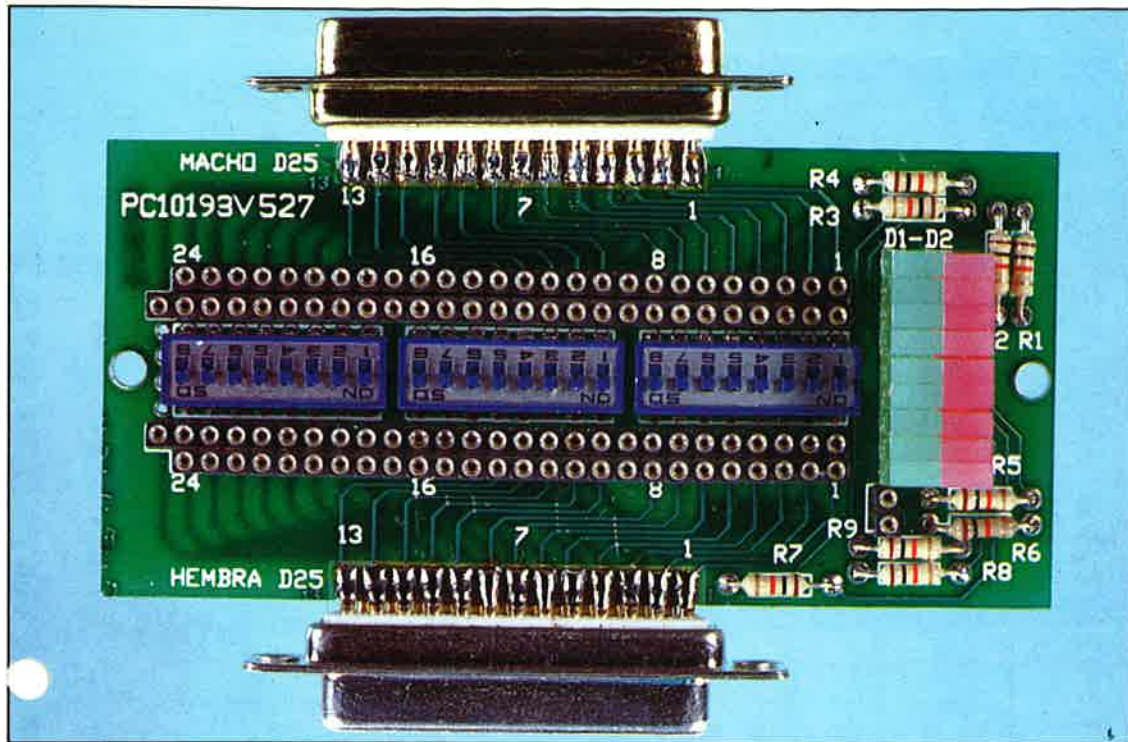
Questa tabella corrisponde a un *codificatore con priorità* 10 a 4 linee. La X indica che l'ingresso può assumere qualsiasi valore, poiché è irrilevante il fatto che sia 0 oppure 1. La disposizione diagonale dei valori 1 dimostra chiaramente come l'attivazione dell'ingresso relativo alla linea 9, quella a priorità maggiore, prevalga sull'attivazione di qualsiasi altro ingresso; infatti, qualunque sia il livello alto o basso (X) presente sugli ingressi da 0 a 8, lo stato delle uscite corrisponde sempre al codice 1001 che è il codice BCD del numero 9. Se si esaminano i diversi valori che assumono le uscite e si tenta di esprimere le equazioni booleane corrispondenti, è possibile notare che queste risultano notevolmente complesse.
Per costruire con delle porte logiche la tabella della verità precedente si devono utilizzare delle porte AND-OR a due e quattro ingressi. Generalmente gli integrati che contengono questo tipo di

*Codificatore con priorità a 8 ingressi e segnale di abilitazione*

logica vengono costruiti in media scala di integrazione MSI, la cui priorità codifica 10 linee decimali in 4 linee BCD.

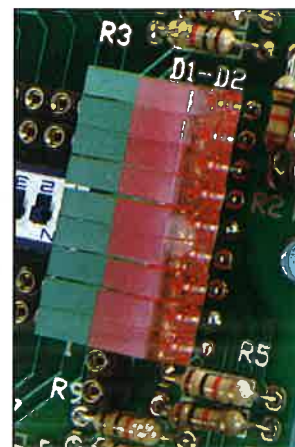
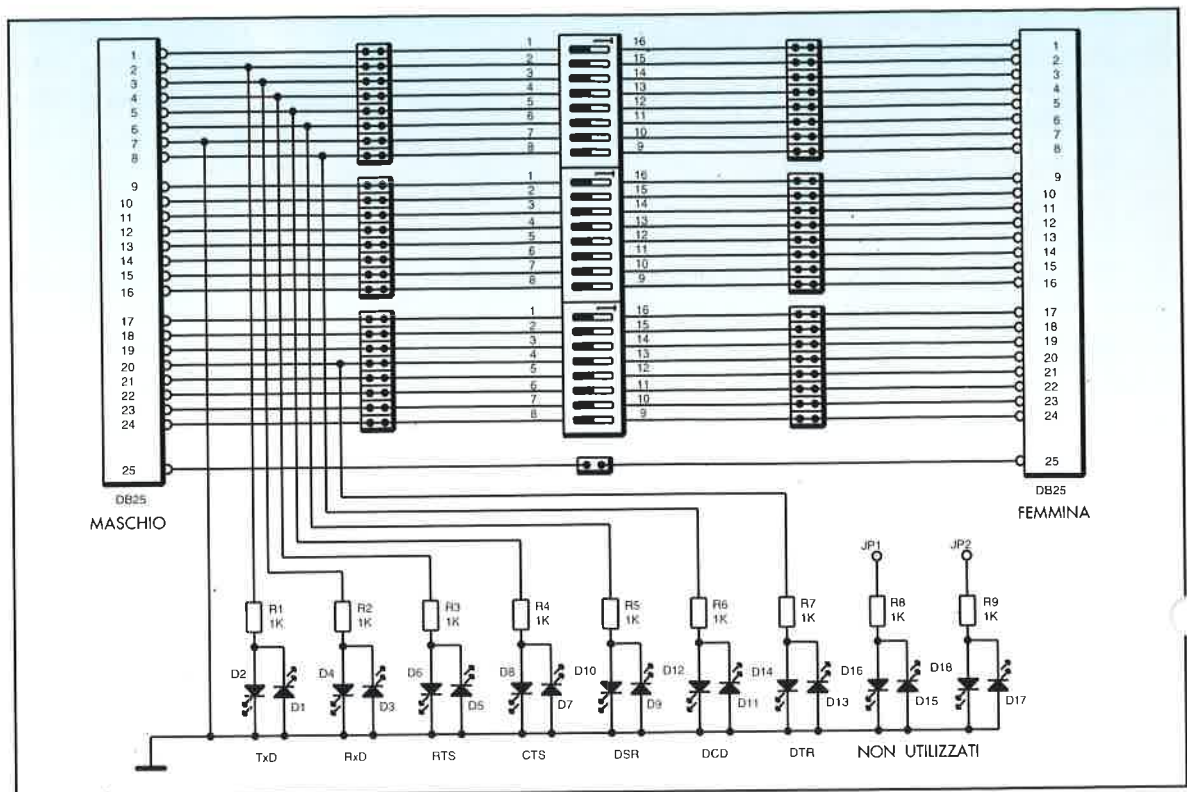
Tra le varie applicazioni di questo dispositivo, le più importanti sono la codificazione di piccole tastiere, la conversione analogico-digitale, e il controllo dei disturbi negli elaboratori.

L'applicazione più significativa però è quella che prevede l'utilizzo di questi codificatori per la realizzazione di particolari dispositivi definiti *multiplexer*, che saranno esaminati nei prossimi capitoli.

*Una delle applicazioni più comuni dei codificatori è costituita dalla conversione tra codici; in questo caso è schematizzato un convertitore da BCD a binario a due decadi*

*Circuito integrato di un codificatore con priorità*

# COMMUTATORE UNIVERSALE RS232

**Per collegare un elaboratore ad un periferica tramite l'uscita seriale generalmente si utilizza il protocollo RS232, che ha raggiunto un livello di diffusione tale che praticamente tutti i calcolatori sono dotati almeno di una porta dati seriale con questo tipo di interfaccia.**

L'obiettivo che si voleva raggiungere con la realizzazione di questo tipo di interfaccia era quello di facilitare le comunicazioni tra un terminale, o *DTE (Data Terminal Equipment)*, e un dispositivo periferico, o *DCE (Data Communications Equipment)*.
Questo protocollo prevede che le comunicazioni vengano gestite dal DTE (calcolatore), che le trasmette ad una periferica DCE collegata (stampanti, modem, ecc.).
Come è facilmente intuibile, questo protocollo permette di attivare una comunicazione tra due dispositivi qualsiasi, qualunque sia la loro natura;

*Grazie al protocollo RS232 è possibile mettere in comunicazione tra di loro due dispositivi qualsiasi*

*Schema elettrico del commutatore universale*

nel caso si colleghino tra di loro due calcolatori però, è necessario stabilire quale di questi deve operare come DTE e come si può configurare l'altro come DCE. Come si vedrà successivamente, tutti questi problemi possono essere risolti utilizzando il circuito di commutazione proposto in queste pagine.

## Collegamenti nel protocollo RS232

La comunicazione tra due apparecchiature eseguita tramite l'interfaccia RS232 è fondata su alcune regole proprie di questo protocollo di comunicazione. Queste regole prevedono una serie di parametri che permettono di definire la forma dei segnali, il modo in cui gestirli, e la direzione nella quale devono essere inviati; se si desidera ottenere una comunicazione intelligibile è necessario impostarli correttamente su entrambe le apparecchiature. Le connessioni previste per l'interfaccia RS232 (comunicazione tra un DTE e un DCE) sono molto semplici: infatti, è sufficiente collegare i terminali corrispondenti dei connettori con dei fili di opportuna lunghezza: il terminale 1 del connettore del DTE al terminale

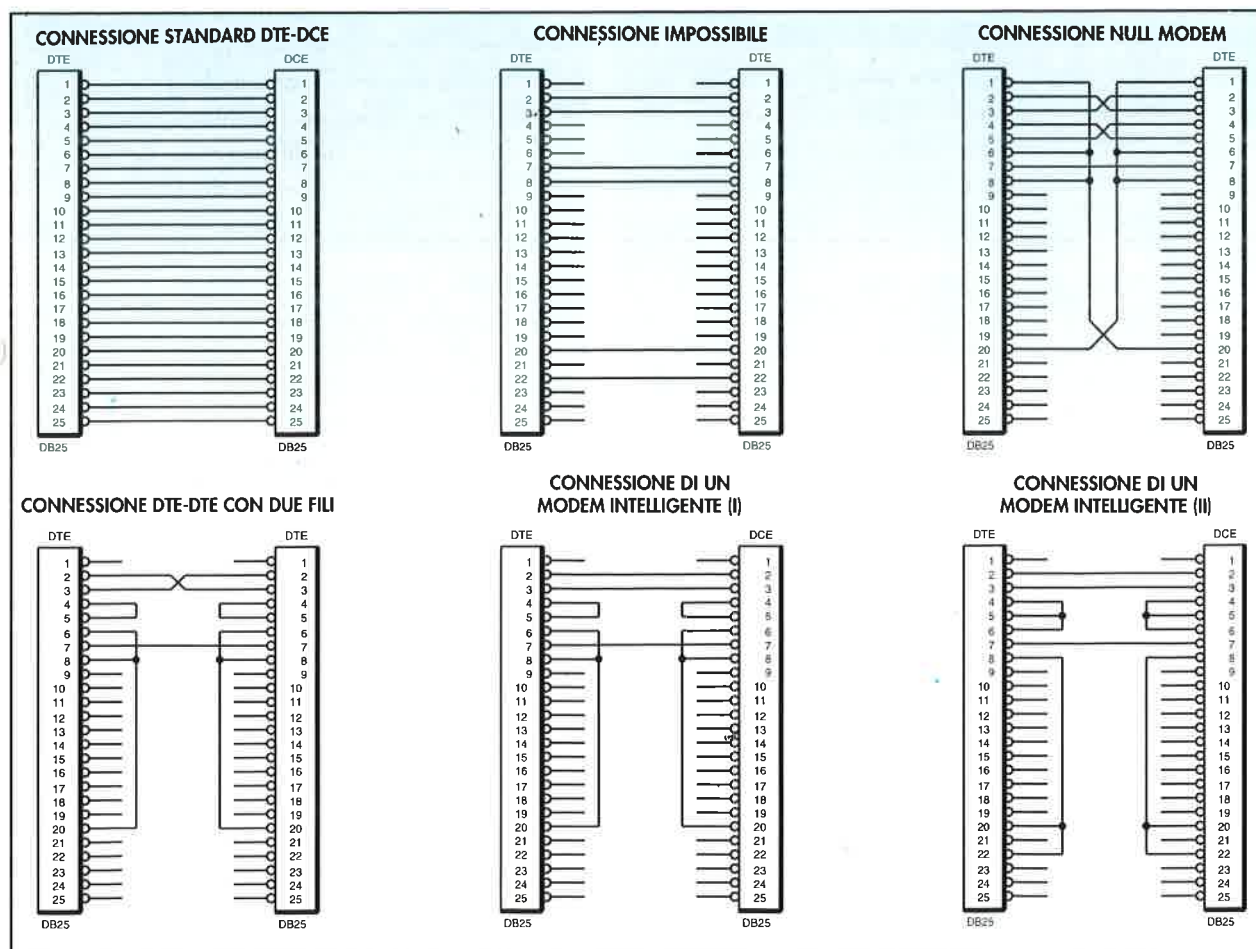
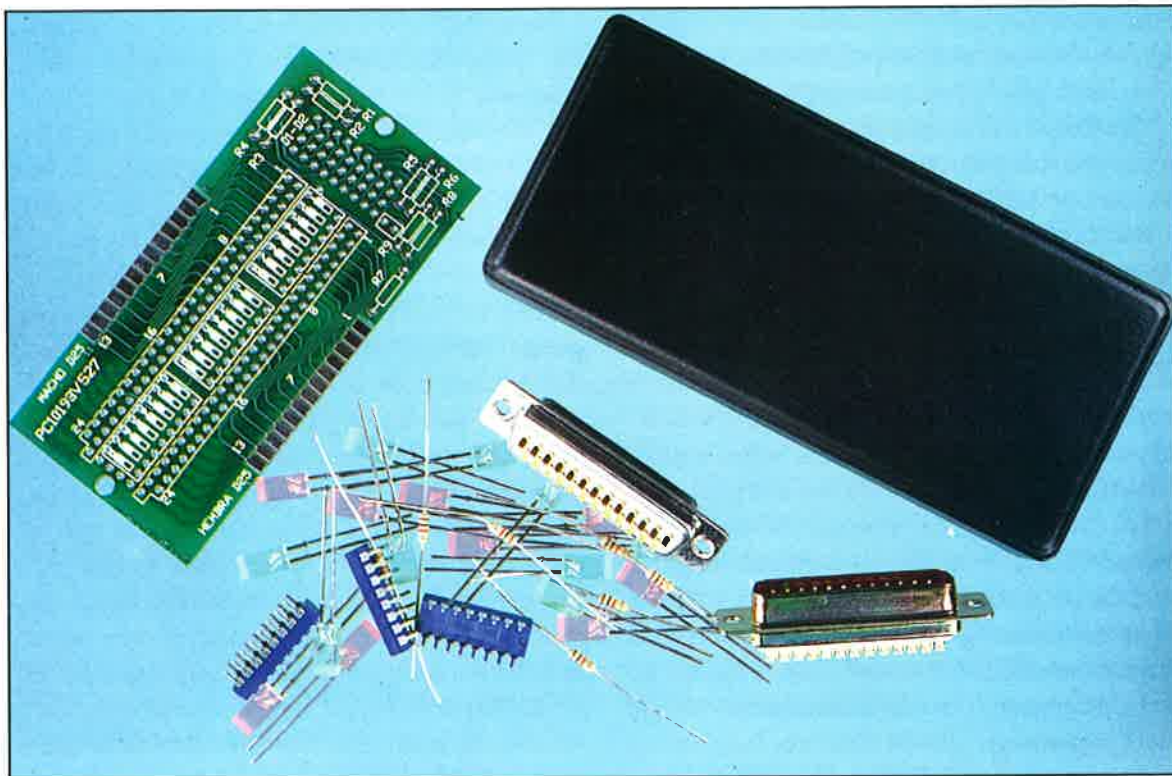
*L'interfaccia RS232 segue alcune norme proprie del protocollo*

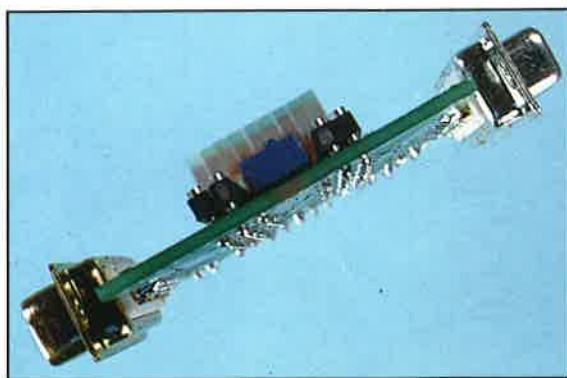
*Distribuzione dei segnali nel connettore RS232*

| PIN | SEGNALE | FUNZIONE DEL SEGNALE |
|---|---|---|
| 1 | CG | MASSA DEL TELAIO |
| 2 | TXD | TRASMISSIONE DATI |
| 3 | RXD | RICEZIONE DATI |
| 4 | RTS | RICHIESTA DI INVIO |
| 5 | CTS | PREPARAZIONE PER LA RICEZIONE |
| 6 | DSR | DATI PRONTI |
| 7 | SG | MASSA DEL SEGNALE |
| 8 | DCD | RILEVAZIONE PORTANTE |
| 9 | V+ | TENSIONE DI PROVA POSITIVA |
| 10 | V- | TENSIONE DI PROVA NEGATIVA |
| 11 | - | NON UTILIZZATO |
| 12 | SDCD | DCD SECONDARIO |
| 13 | SCTS | CTS SECONDARIO |
| 14 | STXD | TXD SECONDARIO |
| 15 | TXC | CLOCK DI TRASMISSIONE |
| 16 | SRDX | RDX SECONDARIO |
| 17 | RXC | CLOCK DI RICEZIONE |
| 18 | - | NON UTILIZZATO |
| 19 | SRTS | RTS SECONDARIO |
| 20 | DTR | TERMINALE PRONTO |
| 21 | SQ | RILEV. DELLA QUALITA' DEL SEGNALE |
| 22 | RI | INDICATORE DI TONO |
| 23 | SEL | SELETTORE DI VELOCITA' DTE |
| 24 | TCK | SELETTORE DI VELOCITA' DCE |
| 25 | BSY | LINEA OCCUPATA |

1 del connettore del DCE, il 2 con il 2 e così via. Il problema, come detto in precedenza, nasce nel momento in cui si devono mettere in comunicazione due DTE; se si esegue il collegamento precedente, per fare in modo che tutto proceda come previsto uno dei DTE dovrebbe essere configurato come DCE. In realtà ciò non è realizzabile, poiché comporta il collegamento degli ingressi di uno con gli ingressi dell'altro e delle uscite del primo con le uscite del secondo, per cui non si può stabilire alcuna comunicazione e non può avvenire in nessun momento uno scambio di informazioni.
Una possibile soluzione potrebbe essere il collegamento dei due computer tramite due modem; in questo caso ciascun DTE (calcolatore) è collegato al suo DCE (modem), per cui si evita di incorrere nella situazione descritta in precedenza. Tuttavia, come molti lettori già sanno, il collegamento di due calcolatori tramite modem è consigliabile ed economicamente plausibile solamente quando questi si trovano a grande distanza. Per distanze brevi la soluzione ideale è il collegamento diretto dei calcolatori tramite cavo, poiché in questo modo si evitano tutti gli inconvenienti legati all'uso dei modem, quali il rumore presente sulle linee telefoniche, la velocità di trasmissione, ecc., per cui aumenta l'affidabilità complessiva del collegamento. Questa condizione si può ottenere modificando il cavo standard per trasmissioni RS232. Il procedimento che si deve seguire per realizzare questa modifica consiste nel collegare gli ingressi di un connettore con le uscite dell'altro e le uscite del primo con gli ingressi del secondo, in modo che risultino interconnessi i terminali complementari del protocollo RS232. Agendo in questo modo si ottiene come risultato un tipo di collegamento definito *null modem*, che si può osservare nello schema riportato nella figura corrispondente. Questo tipo di collegamento emula la funzione dei modem che si sarebbero dovuti utilizzare per collegare tra di loro i due computer, facendo in modo che ciascun calcolatore veda

*Nella configurazione "null modem" gli ingressi di un DTE vengono connessi alle uscite di un altro e viceversa*

*Diversi tipi di collegamenti che si possono eseguire con l'interfaccia RS232*

*Scheda del commutatore e componenti prima del loro assemblaggio*

l'altro come un DCE. La connessione più importante in questo tipo di protocollo è quella relativa ai terminali 2 e 3, che corrispondono rispettivamente al terminale di "trasmissione dati" e a quello di "ricezione dati". Collegandoli come indicato si ottiene che il flusso delle informazioni può avvenire in entrambe le direzioni.
Gli altri collegamenti, eccezion fatta per la linea di massa, servono per i segnali di controllo e di verifica del protocollo RS232. La connessione deve essere eseguita tra i terminali complementari del protocollo, come ad esempio il terminale RTS (*Request To Send* o richiesta per l'invio) di un

*Dettaglio dei connettori DB-25 montati sullo stampato*

*Il collegamento "null modem" viene utilizzato per mettere in comunicazione due elaboratori*

computer con il terminale CTS (*Clear To Send* o cancellazione per l'invio) dell'altro; in questo modo un calcolatore richiede il consenso per inviare i dati (attiva RTS) mentre l'altro riceve il segnale di preparazione per la ricezione.
É possibile realizzare un collegamento tra due DTE anche semplicemente per mezzo di due fili (si osservi lo schema nella figura corrispondente) e, ovviamente, la massa di riferimento. Il funzionamento in questo caso è molto più semplice rispetto a quello del "null modem", poiché è sufficiente eseguire la connessione del terminale di trasmissione di un DTE con quello di ricezione dell'altro DTE. I segnali di controllo del protocollo devono essere eseguiti in modo che ciascun DTE (calcolatore) li invii a se stesso. Questo sistema di comunicazione può generare dei problemi quando la velocità di trasmissione è relativamente alta, poiché i segnali di controllo della trasmissione non vengono forniti dall'elaboratore ricevente, per cui non esiste nessun tipo di mutua informazione tra i due che permetta la correzione degli eventuali inconvenienti che si possono verificare. Questi si possono generare con velocità superiori ai 9.600 bit/secondo. Per collegare un modem intelligente si può utilizzare lo schema "I" di

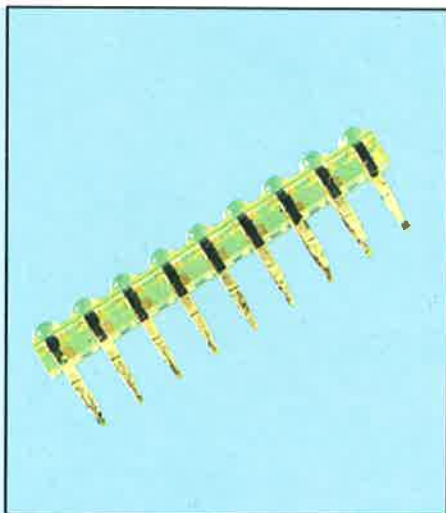
*Dettaglio dei diodi LED*

figura, che corrisponde ad una variante del collegamento tra due DTE realizzato con due fili. La connessione tra il terminale di trasmissione dei dati (TXD) e quello di ricezione (RXD) viene eseguita come se si trattasse di un collegamento tra un DTE ed un DCE (in pratica il modem è il DCE). Questa configurazione può essere applicata perché i modem intelligenti non hanno bisogno dei segnali di protocollo RS232. Tuttavia, se il sistema operativo richiede la presenza di questi segnali, è possibile utilizzare lo schema di collegamento "II", che permette sia l'uso di un modem intelligente che l'inter-scambio dei controlli per gli ingressi e le uscite.
Tutto quanto detto sinora può far pensare che il protocollo RS232 sia molto complesso, a causa della molteplicità delle configurazioni che può assumere; ciò in parte è vero, soprattutto per la rapida espansione delle applicazioni che sfruttano questo protocollo, ed è per questo motivo che si rende necessario semplificare al massimo le diverse configurazioni di collegamento. Questo obiettivo si può ottenere utilizzando il commutatore universale di protocollo RS232 che viene di seguito proposto.

## DESCRIZIONE DEL CIRCUITO

Questo circuito è praticamente un "cavo" di collegamento standard (collegamento tra DTE e DCE) che permette però di modificare la configurazione del collegamento grazie alla presenza di microinterruttori con i quali è possibile abilitare o disabilitare le diverse linee in funzione del tipo di apparecchiatura che deve essere collegata. I "fili" che compongono il cavo sono realizzati dalle piste stesse del circuito stampato (come si può osservare nello schema relativo), che collegano i terminali corrispondenti dei connettori maschio e femmina del circuito commutatore. La continuità di queste piste presenti sul circuito stampato possono essere interrotte tramite i microinterruttori DIL.
Dopo aver interrotto le linee necessarie per mezzo dei dip-switch corrispondenti, è possibile eseguire le connessioni richieste con dei ponticelli inseriti negli appositi terminali torniti femmina presenti alla destra e alla sinistra dei microinterruttori. In questo modo è possibile modificare velocemente e con bassissima probabilità di errore i collegamenti dell'uscita seriale del proprio elaboratore.
Le strisce di terminali femmina sono doppie principalmente per due motivi: per poter collegare due linee allo stesso segnale (come nel caso dell'allacciamento a due fili con i modem intelligenti), oppure per favorire il collegamento di eventuali strumenti di misura. Questi terminali consentono un facile accesso a tutte le 25 linee di trasmissione che arrivano al dispositivo collegato, e una loro gestione indipendente; per bloccare la

*Ingegnerizzazione del coperchio del contenitore*

***Le interconnessioni presenti sul circuito stampato possono essere interrotte tramite i microinterruttori DIL***

*I diodi LED possono essere sostituiti con due array da nove LED*

trasmissione di un segnale è perciò sufficiente interrompere la linea che lo trasporta tramite il microinterruttore corrispondente. In questo modo è possibile monitorare il segnale che si desidera senza considerare l'influenza che il dispositivo collegato può provocare sullo stesso, poiché non è necessario scollegare il circuito commutatore; questa possibilità si rivela molto utile nella ricerca guasti su dispositivi difettosi.

*I diodi LED devono essere collegati a coppie contrapposte su ogni linea utilizzata*

Dopo aver analizzato il funzionamento del circuito principale di commutazione, si può valutare un altra parte del circuito che, anche se meno importante, risulta molto pratica quando si opera con le comunicazioni seriali: quella della visualizzazione dei segnali. Questo circuito è composto da alcuni diodi LED collegati alle linee del circuito di commutazione che servono per indicare lo stato dei principali segnali durante la trasmissione (un colore per il livello alto ed un altro per quello basso). I diodi sono due per ogni linea, e sono collegati in contrapposizione tra di loro. I LED si illuminano in funzione del livello logico assunto da ciascuna linea, e perciò al valore di tensione positivo o negativo applicato alla stessa; grazie a questo sistema uno dei due diodi LED si illumina quando la relativa linea si trova a livello basso, mentre l'altro rimane spento e viceversa.

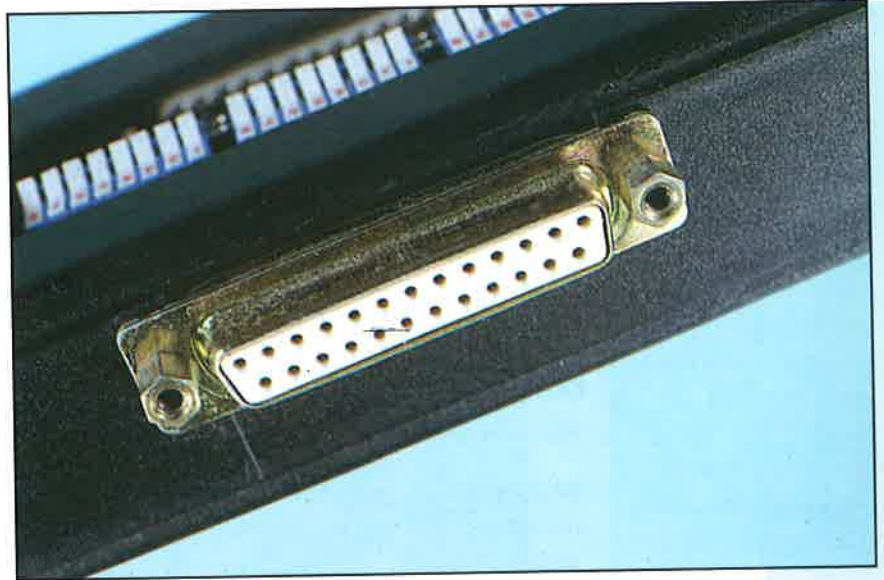
Si sarebbero potuti utilizzare 25x2 indicatori per visualizzare lo stato di tutte le linee, ma questa non sarebbe stata una soluzione opportuna per due ragioni principali: per ottenere un grado di trasmissione affidabile generalmente vengono utilizzate al massimo 7 linee (le 7 collegate), mentre le rimanenti non vengono collegate o non sono indispensabili. Per i lettori più curiosi sono state inserite però due ulteriori coppie di diodi LED, che possono essere collegate ad una qualsiasi delle linee del commutatore per visualizzarne il funzionamento; i terminali liberi di questi diodi corrispondono ai punti JP1 e JP2 riportati sullo schema elettrico. L'altra ragione è ovvia: un commutatore dotato di 50 diodi LED risulterebbe troppo grande e perderebbe una delle sue migliori caratteristiche, la maneggevolezza.

*Le viti di fissaggio del connettore maschio devono essere perfettamente a filo con il contenitore*

## MONTAGGIO DEL CIRCUITO

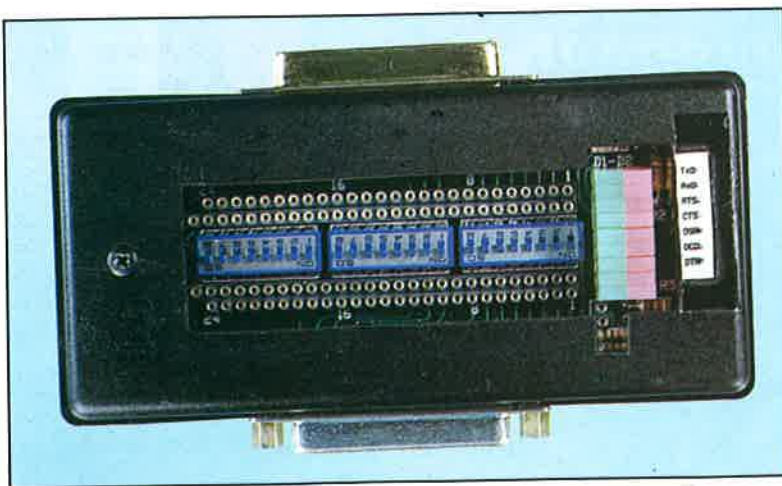
Il circuito, essendo digitale, non richiede alcun tipo di regolazione e il suo montaggio è piuttosto semplice. Il primo fattore che si deve tener presente è che questo circuito è a doppia faccia con fori METALLIZZATI, per cui NON si devono eseguire saldature sul lato componenti.

I primi componenti che devono essere montati sono, come sempre, i componenti passivi, costituiti in questo caso dalle nove resistenze di limitazione R1-R9. Di seguito si possono montare e saldare i microinterruttori DIL e la prima striscia di terminali torniti femmina; dopo aver verificato, ed eventualmente corretto, l'allineamento dei terminali di questa prima striscia si può procedere con il montaggio della seconda. I connettori non sono predisposti per la saldatura su circuito stampato; si sono infatti preferiti quelli con i terminali a saldare, normalmente utilizzati per il collegamento con i cavi, poiché garantiscono una maggiore rigidità al complesso, anche se richiedono certe precauzioni per il loro montaggio. Per prima cosa è necessario verificare che il circuito stampato possa essere inserito tra le due file di terminali dei connettori senza forzare eccessivamente, poiché al contrario si potrebbero danneggiare le piste. Se questo intervento appare difficoltoso bisogna separare i terminali

con un attrezzo e ripetere la prova finché l'inserimento avviene con la necessaria facilità. Al termine di questa operazione è necessario, prima di eseguire le saldature, allineare correttamente i terminali con le rispettive isole rettangolari presenti sul circuito stampato. Inoltre, se si utilizza un contenitore, bisogna anche regolare la distanza di interasse tra i due connettori poiché questi devono fuoriuscire in egual misura dalle corrispondenti fessure per poter essere fissati all'esterno della scatola, come illustrato nella relativa figura. Dopo aver verificato la correttezza di tutte le regolazioni suggerite si possono eseguire le saldature dei terminali. I diodi LED devono essere saldati in modo che risultino alla stessa altezza dei microinterruttori, per evitare che fuoriescano dal contenitore utilizzato. Inoltre, bisogna posizionarli in modo che quelli rossi risultino in opposizione a quelli verdi, per cui l'anodo dei primi deve essere elettricamente collegato al catodo dei secondi. Un'altra considerazione tanto importante quanto ovvia è relativa proprio al colore dei diodi: questi devono essere inseriti in modo che tutti quelli dello stesso colore indichino lo stesso livello poiché, in caso contrario, l'osservazione dei segnali potrebbe risultare piuttosto difficile. Il criterio che viene seguito per la costruzione delle sonde logiche può essere considerato valido anche per il collegamento di questi LED: *colore verde per il livello alto* e *colore rosso per il livello basso*. Se si desiderano utilizzare LED di altro colore è possibile farlo, tenendo presente

*Per il DB-25 femmina si possono utilizzare i distanziali a vite che servono anche per fissare il connettore associato*

però che i LED più comuni e più facilmente reperibili sono sempre quelli rossi e verdi. Se per qualche lettore l'operazione di montaggio dei LED risulta difficoltosa, è possibile utilizzare i diodi *pack*, costituiti da strisce di diodi che si inseriscono perfettamente nelle isole già predisposte sul circuito stampato; questa scelta velocizza il processo di montaggio e saldatura, poiché i diodi di queste strisce sono già tutti alla stessa altezza e i loro terminali sono già alla corretta distanza. Al termine di questa operazione si può considerare il circuito finito; è consigliabile però isolarlo in qualche modo dall'ambiente esterno per evitare che, maneggiandolo, possa subire dei danni. Meccanicamente il dispositivo presenta già una certa rigidità e robustezza, grazie ai connettori utilizzati. Dal punto di vista elettrico invece, una protezione sufficiente potrebbe fornirla una lacca isolante (facilmente reperibile nei negozi di componenti elettronici); se si utilizza questo tipo di lacca però, prima di spruzzarla sul circuito bisogna proteggere con del nastro o della carta adesivi gli zoccoli che servono per l'interconnessione delle linee, poiché se l'isolante riesce a penetrare nel foro del terminale potreb-

*Aspetto finale, decisamente professione, del commutatore universale completo di contenitore*

*I diodi LED devono essere saldati in modo tale da risultare alla stessa altezza dei microinterruttori*

***Elenco componenti***

**Resistenze**

R1-R9 = 1 kΩ, 1/4 W, 5%

**Semiconduttori**

D1,3, 5, 7, 9, 11, 13, 15 e 17= diodi LED rettangolari rossi

D2, 4, 6, 8, 10, 12, 14, 16 e 18 = diodi LED rettangolari verdi (tutti sono sostituibili con due array da 9 LED)

**Varie**

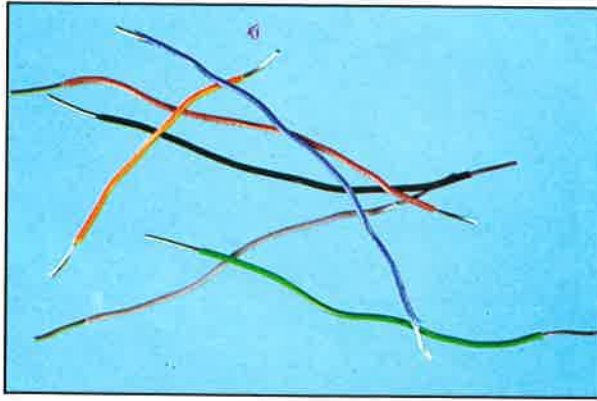
3 dip-switch DIL a 8 posizioni

108 terminali torniti femmina

1 connettore DB-25 maschio a saldare

1 connettore DB-25 femmina a saldare

1 contenitore (opzionale, non compreso nel kit)

1 circuito stampato PC10193V527

*Ponticelli da utilizzare per eseguire i necessari collegamenti volanti*

be provocare dei problemi durante il funzionamento. Un altro sistema di protezione potrebbe essere quello di ricoprire l'intero circuito (con eccezione dei diodi LED e degli zoccoli) con della resina o pasta termoindurente che compatti tutto in un blocco unico. La soluzione più idonea è comunque quella di utilizzare un contenitore plastico. Questo non deve possedere caratteristiche particolari, ma solo delle dimensioni compatibili con quelle del circuito. Sul coperchio superiore è necessario aprire una finestra che consenta l'accesso ai microinterruttori ed ai terminali di interconnessione. Se la larghezza interna del contenitore è appena superiore a quella dello stampato i connettori possono essere saldati direttamente sulla scheda, eseguendo però le opportune regolazioni suggerite in precedenza. Se invece il contenitore è molto più largo dello stampato, si consiglia di collegare i connettori al circuito con dei cavi piatti di lunghezza opportuna. In entrambi i casi è necessario eseguire sulle pareti laterali della scatola le forature richieste per l'inserimento e il fissaggio dei connettori. L'altezza dei diodi LED deve essere tale da non permettere che sporgano oltre il coperchio superiore, ma anche da non crearne l'eccessivo affossamento all'interno della scatola. Se necessario, utilizzare per questi componenti dei terminali torniti. Inoltre, bisogna aprire una finestra in corrispondenza di questi indicatori per permettere la loro visualizzazione dall'esterno. Un esempio di foratura del contenitore è illustrato nelle fotografie presenti in questo capitolo. L'ultima operazione consigliata e sufficientemente importante è quella di applicare una targhetta indicante le linee associate ai diodi LED, come illustrato nella corrispondente figura; in questo modo non è necessario ricordare sempre a memoria quale è il segnale che si sta controllando.

*Dettaglio dei collegamenti "volanti" nel commutatore universale*